# Oque é uma activity?

É um componente do android que representa a tela do nosso aplicativo, onde vamos ter: textos, botões, menus, listas entre outros.

# Uma activity ela é composta por;

Uma class Kotlin e um arquivo XML onde vai ter a parte gráfica.

Os dois juntos vão formar uma acitivity

# Navegando entre activity

Antes de ir para a navegação entre as activity o ideal é entender oque é uma **PILHA** de activity

Uma pilha de activity é; conforme você vai abrindo, uma vai ficando sobre a outra, então o ideal é ir fechando-as para não ocupar muito espaço na memória.

Vamos imaginar que temos 2 activity e você abriu a segunda, então você não precisa voltar para a primeira, você pode apenas fechar a segunda e automaticamente vai retornar para a primeira.

# Como abrir uma nova activity

Para abrir uma nova activity vamos utilizar o método startActivity

Agora para abri precisamos utilizar como parâmetro o objeto **intent**

**startActivity(intent)**

Intent = intenção || (resolve ou faz determinada coisa)

startAcitivity(intente(**contexto**, classQueVaiSerAberta))

**contexto** = contexto que será executado

## Oque é o CONTEXTO

É um ponto de acesso para informações globais do seu App.

Vamos imaginar aqui em casa, tem várias janelas.

- Aí eu falo Fernanda abra a janela (**intente = intenção**)

E aí eu preciso passar o **contexto**

Oque seria o contexto; seria qual janela Fernanda vai abrir, porque tem mais de uma janela

- Fernanda abra a janela(**intente**) da cozinha (**contexto**)

E aí eu passo dessa forma

startActivity(intente(contexto, janelaSegundoActivity::class.java))

## Outro exemplo

startActivity(intent(contexto, JanelaCaronaActivity::class.java))

Vamos imaginar que falo, abra a janela, mas não sabemos qual, então precisamos passar um pouco mais de contexto

Abra janela (intenção)?

Qual? (contexto)

Do carona (JanelaCaronaActivity::class.java)

Resumo a **intent** ela faz algo que você esta mandando, mas ela precisa do contexto para saber onde ela vai fazer.

Então no exemplo de cima a **intent** esta abrindo uma nova Activity, mas ela precisa saber qual (**contexto**) ela vai abrir, ai você usa o uma virgula e passa a activity que precisa ser aberta.

## Abrindo nova Activity na pratica.

class MainActivity : AppCompatActivity() {  
  
 lateinit var btnAbrirSegundaTela : Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** btnAbrirSegundaTela = findViewById(R.id.*button\_abrir*)  
 btnAbrirSegundaTela.setOnClickListener {  
 startActivity(Intent(this, DetalhesActivity::class.*java*))  
 }}  
  
}

Estamos adicionando uma variável que ainda não inicializamos, vamos inicializar depois **lateinit**

Agora estamos transformando a variável em um objeto do tipo **button\_abrir**

E depois estamos fazendo uma função lambda para quando o usuário clicar nesse botão ele abrir uma nova tela.

**setOnClickListener** = um ouvinte que fica esperando ser clicado para disparar um evento, nesse caso o evento vai ser abrir a segunda tela.

This = é para dizer qual contexto vai ser executado no nosso caso, pedimos para ele executar apenas o DetalhesActivity, mas poderíamos pedir para ele executar algo dentro do DetalhesActivity.

## Fechando uma activity

Aqui estamos fechando a segunda activity então ele vai voltar automaticamente para a tela anterior

Não faça isso para a primeira activity se não ele vai fechar o aplicativo.

class DetalhesActivity : AppCompatActivity() {  
  
 lateinit var btnFechar : Button  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_detalhes*)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** btnFechar = findViewById(R.id.*button\_fechar*)  
 btnFechar.setOnClickListener {  
 finish()  
 **}** }  
}